The Hack Chipset and Hardware

Elementary logic gates:
- Nand
- Nor
- And
- Or
- Xor
- Xnor
- Not

Combinational chips:
- HalfAdder
- Adder
- Inc
- ALU

Sequential chips:
- DFF
- Flip
- Register
- RAM
- ROM
- PC

Computer Architecture:
- Memory
- CPU
- Computer

Source: www.nand2tetris.org

CSCI 410
9 – The Computer

The Hack Computer

Main parts of the Hack computer:
- Instruction memory (ROM)
- Memory (RAM):
  - Data memory
  - Screen (memory map)
  - Keyboard (memory map)
- CPU
- Computer (the logic that holds everything together).

Source: www.nand2tetris.org

Instruction Memory

Function:
- The ROM is pre-loaded with a program written in the Hack machine language
- The ROM chip always emits a 16-bit number:
  \[ \text{out} = \text{ROM32K(address)} \]
- This number is interpreted as the current instruction.

Source: www.nand2tetris.org

Data Memory

To read RAM(k):
- set address to k, probe out

To write RAM(k):
- set address to k, set in to x, set load to 1, run the clock

Source: www.nand2tetris.org
The Screen chip has a basic RAM chip functionality:

- **read logic:** \( \text{out} = \text{Screen}[\text{address}] \)
- **write logic:** if load then \( \text{Screen}[\text{address}] = \text{in} \)

**Side effect:** Continuously refreshes a 256 x 512 black-and-white screen device

---

**Keyboard**

- **Keyboard chip:** a single 16-bit register
- **Input:** scan-code (16-bit value) of the currently pressed key, or 0 if no key is pressed
- **Output:** same

**Special keys**:

<table>
<thead>
<tr>
<th>Key Present</th>
<th>Keyboard Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>console</td>
<td>120</td>
</tr>
<tr>
<td>tab</td>
<td>121</td>
</tr>
<tr>
<td>backspace</td>
<td>122</td>
</tr>
<tr>
<td>delete</td>
<td>123</td>
</tr>
<tr>
<td>space</td>
<td>134</td>
</tr>
</tbody>
</table>

---

**Memory**

- Access to any address from 0 to 16,383 results in accessing the RAM16K chip-part
- Access to any address from 16,384 to 24,575 results in accessing the Screen chip-part
- Access to address 24,576 results in accessing the Keyboard chip-part
- Access to any other address is invalid.

---

**CPU - Execute**

- The CPU executes the instruction according to the Hack language specification:
  - The \( D \) and \( A \) values, if they appear in the instruction, are read from (or written to) the respective CPU-resident registers
  - The \( M \) value, if there is one in the instruction’s RHS, is read from \( \text{mem} \)
  - If the instruction’s LHS includes \( M \), then the ALU output is placed in \( \text{al} \), the value of the CPU-resident \( A \) register is placed in \( \text{add} \), and \( \text{write} \) is asserted.

---

**CPU - Fetch**

- Recall that:
  1. The instruction may include a jump directive (expressed as non-zero jump bits)
  2. The ALU emits two control bits, indicating if the ALU output is zero or less than zero

  - If \( \text{reset} = 0 \): use this information (the jump bits and the ALU control bits) as follows:
    - If there should be a jump, the PC is set to the value of \( n \); else, PC is set to PC+1
  - If \( \text{reset} = 1 \): the PC is set to 0. (Restoring the computer)

---

**CPU**

- Instruction memory
- Data memory
- Control signals
- ALU
- CPU register
- Memory
- Load
- PC
- \( \text{inM} \)
- \( \text{outM} \)
- \( \text{writeM} \)
- \( \text{addressM} \)

---

Source:
- www.nand2tetris.org
CPU - Implementation

Chip diagram:
- Includes most of the CPU's execution logic
- The CPU's control logic is hinted; each circled “C” represents one or more control bits, taken from the instruction
- The “decode” bar does not represent a chip, but rather indicates that the instruction bits are decoded somehow.

Source: www.nand2tetris.org

Notes

- Provided for you:
  - ARegister
  - DRegister
  - ROM32K
  - Screen
  - Keyboard
- Also, use built-in chips (no need to copy your implementations, although you could)
- Eat the elephant one bite at a time!

State of the Art

“Real” Computers:
- Caching
- More I/O units
- Special-purpose processors (I/O, graphics, communications, …)
- Multi-core / parallelism
- Efficiency
- Energy consumption considerations
- And more...